Homework Assignment №1

Dr. Duncan, CSC 3102, Louisiana State University Due: 1/23/2017

Naive versus Horner's Polynomial Evaluation Methods

The purpose of the first homework assignment is to familiarize you with the programming standards and coding conventions that are required for this course. All students taking this course are expected to have a certain level of proficiency which includes the ability to read and understand code written by others, write code that efficiently implements and test various algorithms using programming constructs studied in your programming classes and functions whose syntaxes and usage information you can find in the online C++ references.

In this homework assignment and all subsequent programming projects, you will be expected to properly document your code using the coding conventions for this course, program in multiple files so that there is a wall between the public interface and implementation of your algorithms as well as the application that uses the implementation. This assignment will test your ability to follow this approach to programming no matter what IDE (integrated development environment) that you use to write your code. All programs that you write in this course must be ISO C++11-complaint. Your program will be considered acceptable for grading only if it compiles without any syntax errors. You will generally submit only your source code, the .cpp and .h files, in a zip archive file via a drop box on Moodle for grading. This programming exercise will be evaluated under "Homework Assignment" category but all subsequent programs will fall in the "Programming Projects" category.

In this homework assignment, you will write code to empirically compare the performance of Horner's and naive polynomial evaluation methods. See C-1.14, p. 46, for a detailed description of Horner's method. The naive polynomial evaluation method consists of the use of a nested loop: the inner loop, computes the powers and the outer loop sums the terms of the polynomial. I have provided *Polynomial.h* a header file that contains a class that models a polynomial. The class contains prototypes for the Horner's and naive polynomial evaluation functions. The class also has prototypes for two constructors. See the file for the a description for how these functions should work when they are implemented. Do not make any change to this file.

The Polynomial.cpp File

Definition 1. A **univariate polynomial** is a mathematical expression involving a sum of powers in one variable multiplied by coefficients. A polynomial in one variable with constant coefficients is given by the expression $p(x) = \sum_{i=0}^{n} c_i x^i$, where the $c_i's$ are numeric values representing the

coefficients. The polynomial is said to be an n^{th}-*degree* polynomial if its highest power is n. For example, $3x^4 - 2x^3 + x^2 - 1$ is a fourth-degree polynomial.

To evaluate a univariate polynomial, given a numeric value, the value is substituted for the variable and the expression is evaluated. For example, given the polynomial $p(x) = 3x^4 - 2x^3 + x^2 - 1$, $p(-2) = 67$ since $3 \times (-2)^4 - 2 \times (-2)^3 + (-2)^2 - 1 = 67$. A univariate polynomial. can be represented as an array of its coefficients in descending powers. For example, the polynomials $3x^4 - 2x^3 + x^2 - 1$ and $3x^2 + 2x - 5$ are represented as $[3, -2, 1, 0, -1]$ and $[3, 2, -5]$, respectively.

Provide implementations for the member-functions whose prototypes appear in the class in Polynomial.h. See the header file for information on the functions.

The PolynomialDemo.cpp File

This file contains the main function. The main function will perform the following tasks:

1. Create the two polynomials, display their degrees and evaluate them at the values shown in the sample run.
2. Next, your program will generate 10 polynomials of degrees 1,000, 2,000, $\cdots$, 10,000 with randomly generated coefficients in the range [0,5]. For each polynomial, your program will generate a random number in the range [0.5,1.2] and evaluate the polynomial using the Horner's and naive polyomial evalution methods. Your program will display the execution times for each evaluation as shown in the table in the sample run.

.

Additional Requirements

Test the program to ensure that it works correctly and generates its output in the same format as shown in the sample run. Each file should have the following javadocs:

```
/**
 * EXPLAIN THE PURPOSE OF THIS FIlE
 * CSC 3102 Homework Assignment # 1
 * @author YOUR NAME
 * @since DATE THE FILE WAS LAST MODIFIED
 * @see A LIST OF FILES THAT IT DIRECTLY REFERENCES
 */
```

Also, submit an excel spreadsheet, *polynomialevaluatio.xls*, containing the data generated by your program and a line graph of execution times of each algorithm in nanoseconds (Y-axis) versus degrees of the polynomials (X-axis). Locate the source files, *Polynomial.h*, *Polynomial.cpp* and *PolynomialDemo.cpp* and enclose them along with the spreadsheet in a zip file. Name the zip file *YOURPAWSID_hw01.zip*, and submit your homework assignment for grading using the digital dropbox on Moodle.

Sample Run:

```
f := [4, 5, 0, 2, 3, 5, -1]
deg f(x) := ...
Using Horner’s method, f(3) = .....
Using naive method, f(3) = .....

g := [12.5, 0, 0, -1, 7.2, -9.5]
deg g(x) := ...
Using Horner’s method, g(-7.25) = ....
Using naive method, g(-7.25) = ....

Empirical Analysis of Naive vs Horner’s Methods
on 10 Polynomials with Random Coefficients
================================================
Degree    Naive Time (ns)     Horner’s Time(ns)
------------------------------------------------
1000      .........           .........
2000      .........           .........
3000      .........           .........
4000      .........           .........
5000      .........           .........
6000      .........           .........
7000      .........           .........
8000      .........           .........
9000      .........           .........
10000     .........           .........
------------------------------------------------
```